Inference Focus: A survey of AI acceleration architectures for emerging Large Language and Reasoning Models (LLMs)

Katie Adiletta
Computer Architecture - Worcester Polytechnic Institute
kmadiletta@WPI.edu

***Abstract* - The two main stages in an AI model lifecycle include *training* and *inference*. This report will conduct an initial review of Large Language Model (LLM) AI training vs inference. Then, focus on how current processing architectures have evolved to efficiently compute *inference*. The focus on LLMs is due to the popular use of models such as ChatGPT (OpenAI) and the Llama open-source models which have dramatically influenced interactive conversation, programming, and creative content generation. The emerging architectures that can execute these LLMs are sorted into three categories: CPU, GPU, and function specific processors. One processor specifically, Groq's Language Processing Unit, addresses a key issue of memory bandwidth with its Tensor Streaming Processor. It is packed with parallelism and deterministic behavior and will serve as an example of an innovative architecture for high performance inference. This is significant as the bulk of inference computation is currently performed on GPU's and even more specifically on GPUs from Nvidia [1].**

I. Introduction

The foundation of a Large Language Model is built through a computationally extensive process of training. The first phase prior to the foundation is data-preparation which includes data collection and formatting as it cleans and processes a relevant dataset. This dataset must include "labeled" or curated data which training will use to understand what is correct. The training phase involves establishing matrices of "parameters" and then tuning these internal parameters to improve the performance of the model and minimize errors in its output. In order to handle the resources and time required to train the model, Graphical Processing Units (GPUs) and Tensor Processing Units (TPUs) are generally used as they specialize in computation [2]. Given a trained general model, additional "fine tuning" phases may be employed to focus the model on certain capabilities or enhance a response direction. Given a fine-tuned trained model, the next stage in the AI model's lifecycle is inference: using the learned internal parameters to process new inputs and compute a prediction or output [3]. Inference is generally performed in real time (for conversation and response generation; though for non-real time summarization or content creation the same process is employed) where the user provides an input and the model computes the output [2]. While inference is not as computationally extensive as training and can be computed on standard hardware, models with billions of parameters and "tera's" of floating-point additions and multiplications require hardware that will accelerate these computations. That is an important observation – training, with its attention and feedforward deep matrix computation and higher required data precision, is significantly higher computationally intensive than inference because inference is generally limited by *memory bandwidth* [4]. Nvidia has stated that inference efficiency is a 'core limiting factor for overall model intelligence and the viability of agentic pipelines' which has made inference efficiency a primary optimization objective for recent models and architectures [5].

II. Inference Model Architecture

A. Layer Description

Most Transformer-based Large Language Models use an encoder-decoder structure. The encoder maps the input sequence (the incoming user provided "query") and the decoder generates the output (user sought "answer" or "response") with numerous multi-head attention layers in the middle, seen as Nx in Figure 1 [6].

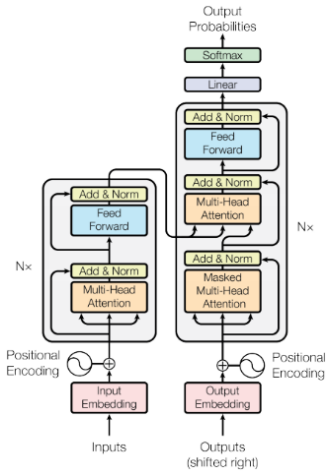

Figure 1: Transformer LLM Architecture [6]

The multi-head attention layers perform scaled dot products between query, key, and value vectors as seen in Figure 2. While the detail of this computation is beyond the scope of this paper, relative to inference computation architecture, it is important to recognize that all these operations require operands (bandwidth) and matrix computation.

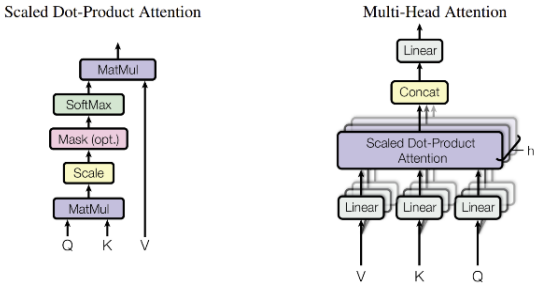

Figure 2: Multi-Head Attention with Scaled Dot-Product Attention [6]

B. Types of Parameters

As previously mentioned, LLMs have millions or billions of parameters to tune and load into memory during training. The data type and precision of the parameters determine how much of the processor's memory the parameters will occupy. Earlier LLMs used the floating-point 32 (FP32)

which takes 32 bits of memory for each parameter. This precision can be problematic as up to 28 GB of expensive memory capacity is required if the model is a 7 billion parameter model. LLMs evolved to reduce the memory capacity "footprint" by using a 16-bit floating-point (FP16) – making LLMs more efficient with parameters that are ½ the original capacity. With this precision and a 7 billion parameter model, the model requires 14 GB of the computer's memory, making it more accessible [7]. A distinct innovation to the data types for the parameters comes from BitNet b1.58 which is used for Ternary LLMs. The idea is that the weights in the transformer layers are ternary and thus use -1, 1, or 0. The weights, therefore, are around 1.58 bits which dramatically reduces the model size. BitNet has a 6.25x speedup over full-precision baselines and it ensures lossless edge inference [8].

Most recently DeepSeek introduced a very accurate model which was the first to be trained using FP16 then distilling inference models to 8bit parameters – again shrinking the memory footprint by a factor of 2. This has a significant cost and bandwidth/performance positive impact. After DeepSeek demonstrated accuracy at this precision, Llama and other models have been gradually moving the inference data precision mainstream to 8bit precision [9].

C. *Inference Stages: Prefill vs Decode*

The process of inference includes computing the first token then the next tokens. The *prefill* stage generates the first output token from an encoded input sequence. The key and value pairs are also cached during the prefill stage also known as the KV cache, so these pairs do not need to be generated again during the decode phase. The prefill stage sets up the model to generate output efficiently, which requires high computation. The decode phase generates the token autoregressively by predicting the next token and updating the KV cache as seen in Figure 3 [10]. The decode phase allows for the next tokens to flow which requires memory bandwidth.

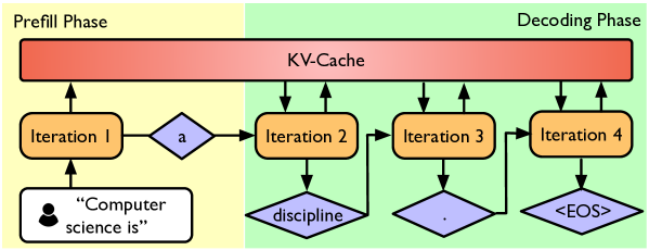

Figure 3: Prefill and Decode Depiction [10]

III. Model Types

A. *The Basics*

When studying different types of LLMs there will either be a standard LLM with basic response capabilities or there will be reasoning models which perform complex reasoning and can "think". Reasoning models answer questions with multi-step generation and intermediate steps as seen in Figure 4 [11].

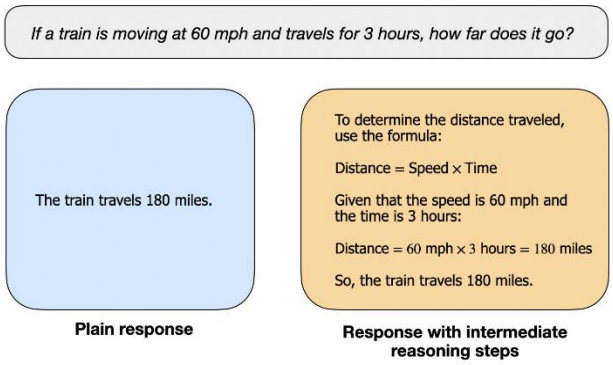
If a train is moving at 60 mph and travels for 3 hours, how far does it go?
The train travels 180 miles.
To determine the distance traveled, use the formula:
Distance = Speed × Time
Given that the speed is 60 mph and the time is 3 hours:
Distance = 60 mph × 3 hours = 180 miles
So, the train travels 180 miles.
Plain response
Response with intermediate reasoning steps

Figure 4: Reasoning Models [11]

While not required for simpler tasks such as summarization or translation, reasoning is effective for complex tasks such as problem solving and coding challenges [11]. Reasoning is using system prompts to generate better query responses and, in the process, it generates *many more decode tokens* which now forces any optimized architecture to focus on decode. For models to be capable of reasoning, these models require additional memory bandwidth.

A prime example of a reasoning model is the DeepSeek architecture, which uses a mixture of expert (MoE) machine learning architecture and was trained using large-scale reinforcement learning [12]. In the prefill stage, DeepSeek utilizes a 32-way Expert Parallelism which enhances computational efficiency. This method ensures each GPU processes the same number of tokens. In the decoding stage, the attention uses a 4-way Tensor Parallelism with Sequence Parallelism combined with the 8-way Data Parallelism to increase throughput and improve the model's efficiency [9]. During the decode stage DeepSeek suggests for optimum inference, 320 GPUs are employed in a pool of MoEs. Another prominent reasoning model is OpenAI's GPT-o4 model, which is an LLM trained with reinforcement learning to perform reasoning [12]. It has a long internal chain of thought before answering and is an excellent problem solver. This reasoning model has reasoning tokens in addition to input output tokens so the model can "think." The structure of this reasoning model can be seen in Figure 5 [13]. The larger reasoning models are slower and more expensive, as they use higher computations in their reasoning. And as can be seen, it generates more decode tokens.

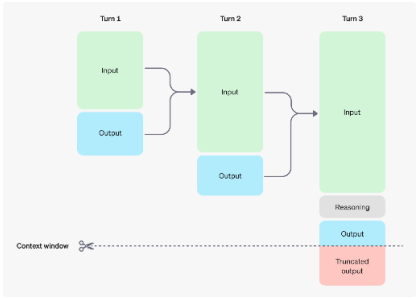

Figure 5: Reasoning Model System – GPT o4 [13]

Recently a family of Llama-Nemotron (LN) models have been announced and are heterogeneous reasoning models. They have inference efficiency and are competitive with

DeepSeek-R1 while fitting on a single 8xH100 node and attaining higher inference throughput. The LN-Super and LN-Ultra models use the *Puzzle* framework which improves computational properties of latency, memory usage, and throughput during inference. The *Puzzle* framework first works by applying block-wise local distillation to construct a library of alternative transformer blocks, then utilizes mixed-integer-programming to create a heterogeneous architecture which optimizes quality under throughput, latency, and memory usage constraints [5].

There is a high diversity of computer architectures that can work to efficiently compute inference to avoid slowdown when using the LLM. These architectures can be split into three categories: Central Processing Unit (CPU), Graphical Processing Unit (GPU), and functional specific processors. In comparison to GPUs, CPUs are lower cost, have fewer cores, are more accessible and widely used, and process data sequentially at good speeds. With these characteristics, CPUs can manage the intensive machine learning workloads. Some examples of standout CPUs include Intel's Xeon CPU Max Series and Tenstorrent's RISC-V TT-Ascalon X.

B. Inference Computation Engines: CPUs

1) Xeon

Intel's Xeon CPU Max Series is supercharged with high bandwidth memory (HBM) to accelerate data-intensive workloads such as modeling or AI. There are up to 56 performance cores using Intel's embedded multi-die interconnect bridge (EMIB) technology and 64 GB of HBM to handle high performance computing workloads. This Xeon series provides up to 20x performance speed-up on Numenta AI technology compared to other CPUs for natural language processing [14]. Key features of Xeon include the Intel Advanced Matrix Extensions (Intel AMX) which introduces eight 2-dimensional registers, or tiles (1 KB in size), where accelerators can operate. It also introduces an accelerator engine, Tile Matrix Multiplication component (TMUL), which is attached to the tiles performing matrix multiplies for AI, as seen in Figure 6, and can perform 1024 FP16 operations / cycle. This extension adds support for up to half-precision floating-point numbers and can deliver up to 5.7x-10x higher generation-to-generation real-time inference performance. This allows Xeon to optimize for deep learning training and inference workloads [15].

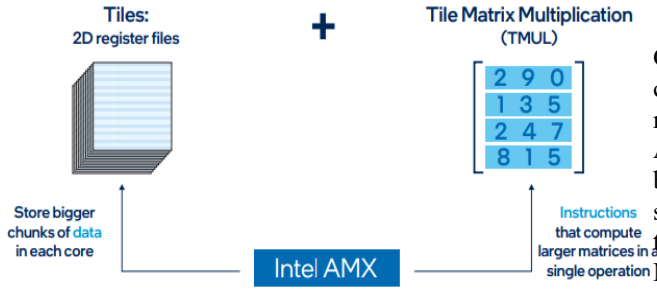

Figure 6: Intel AMX Architecture consisting of 2D register files and **TMUL [15]**

Another component of the Xeon architecture is Intel's Advanced Vector Extensions which accelerates performance of intensive workloads such as AI and deep learning (ie. SoftMax) with vectorization. This feature consists of 512-bit extensions to the 256-bit Advanced Vector Extensions SIMD [16]. With a combination of the Intel AMX, Intel AVX-512, and Intel Data Streaming Accelerator, and an I/O and memory subsystem, this Xeon processor achieves high performance at inference for AI models and can handle extensive workloads.

2) Tenstorrent

Tenstorrent's TT-Ascalon X is optimized to provide high performance while maintaining area and power efficiency. This solution for high performance compute such as AI and ML uses an 8-wide decode unit, has a large L1 cache and shared configurable L2 cache, and uses the Tenstorrent Tensix systolic matrix accelerator IP [17].

C. Inference Computation Engines: GPUs

The next category of computer architectures, GPUs, are known to break down significant problems into smaller ones that can be computed in parallel, therefore providing speed and efficiency for machine learning tasks such as inference. Structurally, a GPU will optimize parallel processing while CPUs optimize sequential processing and in cases where large tasks need to be computed such as LLM inference, parallel computing is attractive. The growth of innovation in GPUs to support AI training and inference has led to even greater opportunities with machine learning and AI and enables more advanced LLMs such as reasoning models.

1) Nvidia H100, H200 etc..

A prominent AI-performance-elevating GPU is Nvidia's H100 Tensor Core GPU which optimizes high-performance computing, processing power, efficiency, and security. This architecture consists of numerous key features such as fourth-generation tensor cores which support data types like FP8 for enhancing deep learning performance. It has enhanced processing rates (3x the A100) and a thread block cluster feature for optimizing parallel processing. The H100 also introduces the first GPU with HBM3 memory translating to a 3 TB/sec memory bandwidth. And it has a Transformer Engine to accelerate AI calculations on transformer models such as LLMs resulting in significant increase in throughput. The Nvidia H100 has been pivotal in high-performance computing and key to the growth of AI as it supports intensive workloads with robust and efficient parallel computing [18].

2) AMD 200, 300

Competitive with Nvidia's H100 in high performance computing and AI is AMD's Instinct MI300 series microarchitecture. The MI300 integrates 8 vertically stacked Accelerator Complex Dies (XCD), 8 stacks of high-bandwidth memory 3, and 4 I/O dies. The Matrix cores are specialized for AI and machine learning by increasing their throughput for existing and emerging data types [19]. The MI300X AI GPU has up to 192 GB of HBM3 and offers bandwidths that support large datasets and AI models. That memory bandwidth enables efficient training and inference of these models. It also achieves up to 153 teraflops for FP32 computations and maintains energy efficiency with advanced thermal control and power management. In addition to GPU's emphasis on AI model training with their

high memory bandwidth and power efficiency, they are great for inference of generative AI models and LLMs with their substantial compute power [20].

D. Inference Computation Engines: Function Specific

The final category of processors are function-specific processors which are designed to optimize for a particular algorithm, workload, or domain. Instead of emphasizing flexibility, it may specialize in speed, power efficiency, throughput, or latency reduction. The following architectures surveyed are designed to optimize for training and inference of AI with high performance of large workloads and efficient parallel computing with low latency and power consumption.

1) Google TPU

Google's seventh-generation Tensor Processing Unit is a scalable custom AI accelerator and is designed specifically for inference. This TPU, Ironwood, prioritizes the demands of *thinking models* such as LLMs, Mixture of Experts and reasoning models. By limiting data movement and latency on the chip and carrying out massive tensor multiplications, Ironwood can carry out the model's massive computational demands. Google focuses on power efficiency and delivers significantly more capacity per watt for customer workloads by utilizing advanced liquid cooling and reliable chip design for high performance. Ironwood is characterized by its increase in high bandwidth memory (HBM) with 192 GB per chip and reaching 7.37 TB/sec per chip to process large models and datasets and ensure rapid data access crucial for AI technology. Ironwood is a unique functional-specific processor that trades generality for computing inference with its computational power, memory capacity, and reliability to serve large workloads with high performance and low latency [22].

2) Gaudi (Intel)

Designed with similar goals of optimizing for the demands of AI applications, the functional-specific Intel Gaudi 3 AI accelerator excels at training and inference through its high compute capability, memory bandwidth, and architectural efficiency. Gaudi 3 has two main compute engines, the Matrix Multiplication Engine (MME) which computes the matrix multiplications of fully connected layers and convolutions, and the Tensor Processor Core (TPC) which accelerates all non-GEMM operations. The architecture consists of two identical compute dies with a high-bandwidth and low latency interconnect as seen in Figure 7. Gaudi 3 has 3.7 TB/sec HBM bandwidth and 128 GB of HBM2e capacity. This architecture is designed to activate all engines (MME, TPC, and NIC) in parallel, allowing for high throughput and low latency [23].

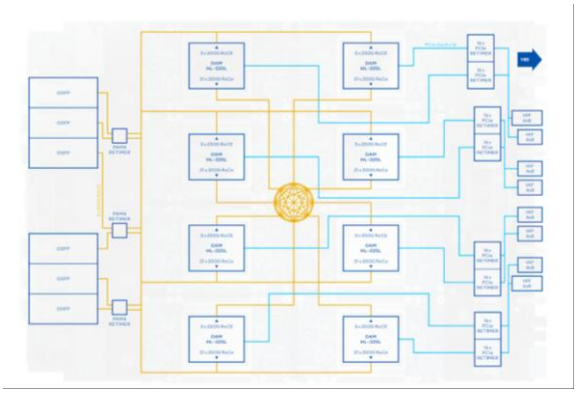

Figure 7: Intel Gaudi 3 Architecture

3) SambaNova

SambaNova stands out as a new processing architecture with its Reconfigurable Dataflow Architecture - a unique vertically integrated platform to accelerate computation. This architecture pipelines operations enabling programmable data access patterns and minimizing excess data movement. Structurally, this architecture is composed of three components. The first is the SambaNova Reconfigurable Dataflow Unit that enables highly parallelizable patterns. Next is the SambaFlow to take input from machine-learning platforms such as Pytorch and TensorFlow. Lastly, the SambaNova Systems DataScale provides an accelerated computing system. Machine learning algorithms are known to have high dataflow-oriented workloads as intermediate results are stored in memory to read and write, as seen in Figure 8 below. This causes large amounts of data movement and memory bandwidth [24]. By handling large dataflow-oriented workloads, SambaNova exemplifies a unique learning system optimized from algorithm to silicon for machine learning, training, and inference.

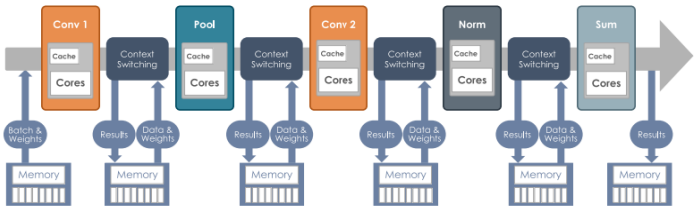

Figure 8: Core-based Kernel by Kernel Execution [24]

4) Cerebras

Cerebras Wafer-Scale Cluster (WSC) is another architecture optimized to handle the massive computational needs of AI model training and memory bandwidth of inference. Powered by the world's largest Wafer-Scale Engine and 125 petaflops of AI compute, it consists of on-chip memory on every compute core with high-memory bandwidth for AI inference. Multiple WSE's are interconnected with on-chip fabric bandwidth and low latency networks. Cerebras also uses a weight streaming architecture to separate the storage of parameters from compute units using the novel MemoryX service which can support up to 120 trillion parameters. It also utilizes interconnection SwarmX fabric which is composed of broadcast-reduce nodes with 100Gb/s of network interfaces [25].

5) Inferentia (Amazon)

Not only does the AWS Inferentia architecture optimize for high throughput and low latency inference, it also looks to reduce costs of inference while maintaining high performance. This specialized AI inference chip handles AI deep learning workloads efficiently and includes 384 GB of shared accelerator memory and offers up to 2.3 petaflops of computing. It integrates other advanced optimizations such as dynamic input shape support, custom operators, and stochastic rounding for performance and accuracy enhancement. The architecture consists of four NeuronCore-v1 cores for 128 INT8 TOPS and 64 FP16/BF16 TFLOPS, 8 GB of DRAM memory with 50 GB/sec of bandwidth and uses Neuron Core Pipeline technology to enable co-optimization of latency and throughput, as seen in Figure 9 below. Notably, these chips are optimized for power efficiency and are usually not always turned on unlike many GPUs, enabling lower electricity costs at high performance [26].

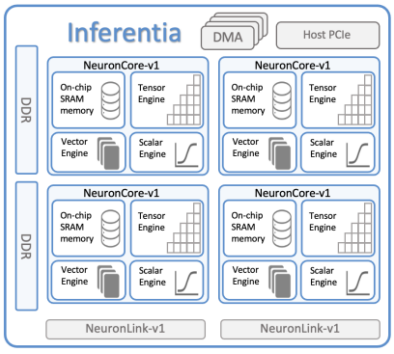

Figure 9: Inferentia Architecture [27]

6) Maia 100 (Microsoft)

Microsoft's Maia 100 accelerator is also specialized for modern machine learning with custom server boards, racks, and software stack to increase performance and reduce costs. The architecture consists of a high-speed tensor for rapid processing of a wide range of data types, a loosely coupled superscalar engine to handle data types such as FP32 and BF16, Direct Memory Access (DMA) engine, and hardware to enable asynchronous programming as seen in Figure 10 below [28].

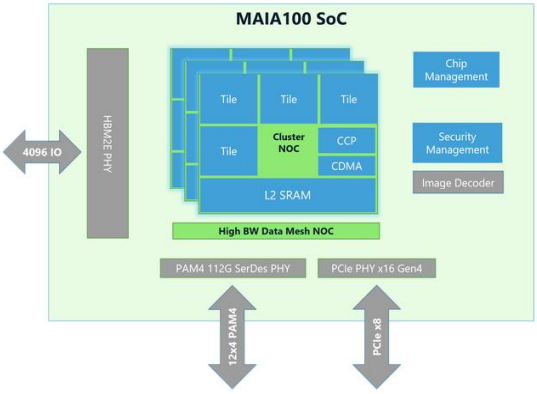

Figure 10: Maia 100 SoC Architecture [28]

Maia optimizes data flow by leveraging SRAM by fusing the post-GEMM activation function to it thus increasing processing speeds and reducing high bandwidth memory reads and reducing latency. By also supporting Pytorch for optimal developer experience, Maia targets AI workloads and parallelizes computations across clusters for efficiency inference computation [28].

7) Groq

Finally, an architecture that deserves a thorough representation is the Groq AI Inference architecture. Groq completely rethinks how processors execute programs. Especially for AI inference workloads as it has broken LLM inference benchmarks with its language processing unit hardware (LPU) on the Tensor Streaming Processor (TSP). This architecture focuses on deterministic and predictable execution through simplified hardware design so the compiler can precisely schedule instructions and data flow. There is massive parallelism with the LPU and a focus on 'computational density' with the TSP making it better suited for Large Language Models.

An interesting design approach in Groq is their Tensor Streaming Processor (TSP), which flips the conventional CPU or GPU design inside out with the functional units outside the cores and arranges them in a 2D grid fashion as seen in Figure 11 below. The functional slices of a TSP are the MXM which performs matrix operations, SXM for shift and rotate operations on vectors, MEM for memory reading and writing, VXM for arithmetic operations on vectors, and the Instruction Control Unit (ICU) for fetching and dispatching instructions. This method of a data-flow architecture puts a focus on movement and transformation of data, whereas the traditional approach uses instructions to determine the data flow [29]. In comparison to other architectures, the TSP has similar qualities to a VLIW as it shifts the complexity of scheduling instructions to the compiler and its deterministic execution for parallel processing.

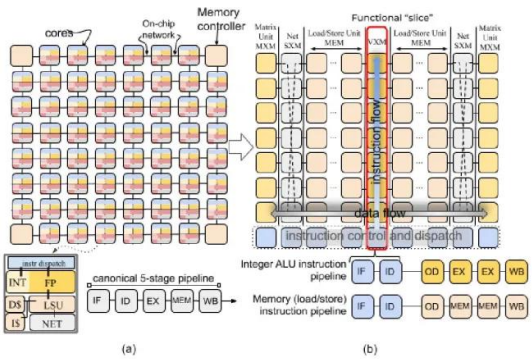

Figure 11: Conventional Multicore chip (left) and flipped design of TSP (right) [29]

The TSP uses simplified hardware and offloads many complexities with the instruction execution into the compiler. The compiler is responsible for scheduling instructions and data flow for a given program and knowing the latency of each instruction, leading to deterministic behaviors in the TSP hardware. The compiler and TSP work together to identify dependencies and assign tasks to available functional units for parallel execution. Since the compiler indicates how the TSP will operate, it is considered a "software-defined hardware" and excels at parallel

computing with its deterministic execution. When all the execution planning happens in software, valuable silicon area is open to provide greater memory bandwidth and transistors for improved performance. Groq also removes cores and hardware that are not specific for AI or machine learning [30].

Overall Groq has a super high performance, delivering ~300-500 tokens/sec while the best GPUs range from around 60-100 tokens/sec. The latency per token for Groq is around 1-2ms while Nvidia has around 8-10ms per token. Groq is also specialized for a batch size of 1 which is largely beneficial for inference compared to Nvidia which needs larger batch sizes. Additionally, Nvidia's H100 costs around $25,000 and it consumes 700W while Groq costs $20,000 and consumes 240W, putting Groq at an advantage over Nvidia [34][35]. Keeping this in mind, Groq is not trying to win the AI market, they are focusing on a specific aspect of AI – inference – thereby targeting where Nvidia underperforms [1].

An additional architecture approach to inference is relying only on SRAM for memory and splitting large models on networked cards. This inference strategy provides advantages like faster access times compared to DRAM, which is crucial for high inference speeds. Also, the bandwidth of SRAM is around 80 TB/sec while the bandwidth of the most efficient HMB3 is 3 TB/sec making SRAM easily the better option for AI inference. However, SRAM is much more costly than DRAM with higher power consumption and has a lower capacity, which means models need to be partitioned across multiple interconnected cards [31]. It is also important to note that there is around 230 MB of SRAM in a Groq AI inference chip [32] while there is around 80 GB of HBM3 in Nvidia's H200 [33]. For Groq to have the same amount of memory as Nvidia it would need 356 Groq chips – a crucial disadvantage to the Groq design. While Groq has impressive speeds for AI inference specific tasks, it trades off cost, power consumption, and scalability. Additionally, unlike GPUs and CPUs, Groq's AI inference architecture is not a general-purpose AI accelerator, so it is only suitable for inference, not training or flexible model research workflows.

Pros:

- Radical architecture shift: Groq rethinks traditional CPU/GPU architecture with its TSP and LPU.
- Software-defined hardware: Offloads scheduling and data flow control to the compiler, allowing deterministic execution.
- Massive parallelism: Functional units arranged in a 2D grid for optimized parallel execution.
- High inference performance: Achieves 300–500 tokens/sec and 1–2 ms/token latency, far surpassing top GPUs like Nvidia H100.
- Optimized for batch size = 1, ideal for real-time, low-latency inference.
- SRAM-based memory: Prioritizes speed and bandwidth over capacity (80 TB/s bandwidth vs HBM3's 3 TB/s).

Cons:

- Not general-purpose: Only suitable for inference, not training or flexible AI tasks.
- High cost and power consumption from SRAM.
- Low memory capacity: Only ~230 MB SRAM per chip vs. 80 GB HBM3 in Nvidia H200.
- Scalability issues: Would require ~356 Groq chips to match the memory of a single Nvidia H200.
- Model partitioning required: Due to limited on-chip memory, large models must be split across multiple chips.

D. Conclusion

This work introduces the problem of computing inference for Large Language models and describes recent advances that optimize for inference in CPUs, GPUs, and function-specific processors. First, however, it is important to identify the stages in the AI lifecycle – specifically training and inference. The training stage optimizes the internal parameters of the model to teach a model to do a task. The computational cost is very high and requires significant GPU/TPU resources. Comparatively, inference makes a forward pass of the model to compute predictions on unseen data – applying what it has learned. The computation is significantly lower than training but still significant, especially on models with huge numbers of parameters. Important characteristics when discussing inference are what data types the architecture uses: FP32, FP16, FP8, or ternary. The next is how it computes prefill, the first token, and decode, the following tokens. Next, there are many LLM models for which the computer architecture specializes, such as reasoning models like DeepSeek ChatGPT o4, or Llama-Nemotron models. Diving into the different architectures, the notable CPU advances for supporting inference include Intel's Xeon or Tenstorrent's TT-Ascalon X. For GPUs, Nvidia's H100 has demonstrated powerful acceleration for both training and inference and is used by OpenAI, Google, Meta, and Microsoft for AI systems. The AMD also has a strong hand in the market with their AMD's Instinct MI300 series. Finally, there have been many recent advancements in functional-specific processors such as, Googles TPU, Gaudi from Intel, SambaNova, Cerebras, and Inferentia from Amazon, Maia from Microsoft. Lastly, the most interesting microarchitecture capturing the focus of this paper is the Groq AI inference Architecture, which utilizes a Tensor Streaming Processor and has a 'software-defined hardware' for deterministic execution. This architecture replaces high memory bandwidth (HMB) with SRAM for incredible inference speeds. Groq's drawback is that using SRAM is costly and it has lower capacity compared to HBM, which results in more chips being necessary for inference of LLMs, perhaps even limiting it to smaller LLM models. This architecture is also not for general-purpose AI as it is specific for inference compared to other powerful processors such as Nvidia's H100. A deeper analysis determines that while the bandwidth of SRAM is 80x that of HBM, the effective performance of tokens/sec only increases by 8x. This presents a leading question of: are the additional 356 Groq chips to match the bandwidth of HBM worth only an 8x performance increase?

REFERENCES

[1] Prathisht Aiyappa, "Groq vs. Nvidia: The Real-World Strategy Behind Beating a $2 Trillion Giant," *Medium*, May 04, 2025. https://blog.startupstash.com/groq-vs-nvidia-the-real-world-strategy-behind-beating-a-2-trillion-giant-58099cafb602.

[2] Tahir, "Generative AI in Action: How Training and Inference Power LLMs," *Medium*, Mar. 12, 2025. https://medium.com/@tahirbalarabe2/generative-ai-in-action-how-training-and-inference-power-llms-6bf9b04a133d

[3] "Understanding LLMs: A Comprehensive Overview from Training to Inference," *Arxiv.org*, 2024. https://arxiv.org/html/2401.02038v1

[4] A. Gholami, Z. Yao, and S. Kim, "AI and Memory Wall," *Arxiv.org*, Mar. 2024. https://arxiv.org/html/2403.14123v1.

[5] A. Bercovich *et al.*, "Llama-Nemotron: Efficient Reasoning Models," *arXiv.org*, 2025. https://arxiv.org/abs/2505.00949.

[6] A. Vaswani *et al.*, "Attention Is All You Need," Jun. 2017. Accessed: Jul. 27, 2025. [Online]. Available: https://arxiv.org/pdf/1706.03762

[7] ApX Machine Learning, "Data Types and Precision (FP16, INT8)," *Apxml.com*, 2025. https://apxml.com/courses/llm-model-sizes-hardware/chapter-3-model-size-hardware-connection/data-types-precision.

[8] J. Wang *et al.*, "Bitnet.cpp: Efficient Edge Inference for Ternary LLMs," *arXiv.org*, 2025. https://arxiv.org/abs/2502.11880.

[9] research@deepseek.com, "DeepSeek-V3 Technical Report," 2024. Accessed: Jul. 27, 2025. [Online]. Available: https://arxiv.org/pdf/2412.19437

[10] B. Li, V. Gadepally, and D. Tiwari, "LLM Inference Serving: Survey of Recent Advances and Opportunities," *Arxiv.org*, 2023. https://arxiv.org/html/2407.12391v1.

[11] S. Raschka, "Understanding Reasoning LLMs," *Sebastianraschka.com*, Feb. 05, 2025. https://magazine.sebastianraschka.com/p/understanding-reasoning-llms.

[12] IBM, "A list of large language models," *Ibm.com*, Mar. 10, 2025. https://www.ibm.com/think/topics/large-language-models-list.

[13] OpenAI, "Reasoning Models," *Openai.com*, 2025. https://platform.openai.com/docs/guides/reasoning?api-mode=responses.

[14] Intel Corporation, "Intel® Xeon® CPU Max Series Product Brief," *Intel® Xeon® CPU Max Series*, 2023. https://www.intel.com/content/www/us/en/content-details/765366/intel-xeon-cpu-max-series-product-brief.html.

[15] Intel Corporation, "Accelerate Artificial Intelligence Workloads with Intel® Advanced Matrix Extensions," *Intel*, 2025. https://www.intel.com/content/www/us/en/content-details/785250/accelerate-artificial-intelligence-workloads-with-intel-advanced-matrix-extensions.html.

[16] Intel Corporation, "Intel® Advanced Vector Extensions 512 (Intel® AVX-512) Overview," *Intel*, 2024. https://www.intel.com/content/www/us/en/architecture-and-technology/avx-512-overview.html.

[17] Tenstorrent, "TT-Ascalon X RISC-V CPU IP." Accessed: Jul. 27, 2025. [Online]. Available: https://cdn.sanity.io/files/jpb4ed5r/production/96c0572a36ab7211bce86d1943aed9719654910d.pdf

[18] J. Gauthier, "High-Performance Computing with the Nvidia H100," *Arkane Cloud*, Jun. 24, 2024. https://arkanecloud.com/high-performance-computing-with-the-nvidia-h100/?utm_source=chatgpt.com.

[19] AMD, "AMD Instinct™ MI300 series microarchitecture," *Amd.com*, 2025. https://rocm.docs.amd.com/en/latest/conceptual/gpu-arch/mi300.html.

[20] server-parts.eu server-parts.eu, "Everything you need to know about the AMD Instinct MI300 GPU," *server-parts.eu*, Oct. 26, 2024. https://www.server-parts.eu/post/amd-instinct-mi300-gpu-ai-hpc-cloud-computing-performance.

[21] A. Cho, "Transformer Explainer: LLM Transformer Model Visually Explained," *Github.io*, 2017. https://poloclub.github.io/transformer-explainer/

[22] A. Vahdat, "Ironwood: The first Google TPU for the age of inference," *Google*, Apr. 09, 2025. https://blog.google/products/google-cloud/ironwood-tpu-age-of-inference/

[23] Intel Corporation, "Intel® Gaudi® 3 AI Accelerator White Paper," *Intel*, Jul. 14, 2025. https://www.intel.com/content/www/us/en/content-details/817486/intel-gaudi-3-ai-accelerator-white-paper.html.

[24] SambaNova, "Resources | White Paper," *Sambanova.ai*, 2024. https://sambanova.ai/resources/tag/white-paper.

[25] S. Hall, R. Schreiber, and S. Lie, "Training Giant Neural Networks Using Weight Streaming on Cerebras Wafer-Scale Clusters," 2023. [Online]. Available: https://8968533.fs1.hubspotusercontent-na1.net/hubfs/8968533/Virtual%20Booth%20Docs/CS%20Weight%20Streaming%20White%20Paper.pdf

[26] N. S. Gill, "AWS Inferentia: Optimizing Multimodal Model Inference Costs," *Xenonstack.com*, Feb. 19, 2025. https://www.xenonstack.com/blog/aws-inferentia-for-multimodal-inference.

[27] AWS, "Inferentia Architecture," *Readthedocs-hosted.com*, 2025. https://awsdocs-neuron.readthedocs-hosted.com/en/latest/general/arch/neuron-hardware/inferentia.html.

[28] S. Xu, "Inside Maia 100: Revolutionizing AI Workloads with Microsoft's Custom AI Accelerator," *Tech Community*, Aug. 27, 2024. https://techcommunity.microsoft.com/blog/azureinfrastructureblog/inside-maia-100-revolutionizing-ai-workloads-with-microsofts-custom-ai-accelerat/4229118.

[29] D. Abts *et al.*, "Think Fast: A Tensor Streaming Processor (TSP) for Accelerating Deep Learning Workloads," May 2020, doi: https://doi.org/10.1109/isca45697.2020.00023.

[30] A. Upadhyay, "The Architecture of Groq's LPU," *Codingconfessions.com*, Mar. 01, 2024. https://blog.codingconfessions.com/p/groq-lpu-design.

[31] J. Elkington, "Groq's inference strategy," *Substack.com*, Oct. 22, 2024. https://axial.substack.com/p/groqs-inference-strategy.

[32] T. Becker, "The Groq AI Inference Accelerator," Nov. 2023. Accessed: Jul. 27, 2025. [Online]. Available: https://sc23.supercomputing.org/proceedings/exhibitor_forum/exhibitor_forum_files/exforum135s2-file2.pdf

[33] Nvidia, "NVIDIA H100 GPU Datasheet," *NVIDIA*, 2022. https://resources.nvidia.com/en-us-hopper-architecture/nvidia-tensor-core-gpu-datasheet.

[34] A. Bayoumi, "Groq AI Chips: A Comparative Analysis of Processing Performance, Cost, and Power Consumption," *Aiixx.ai*, Apr. 30, 2024. https://aiixx.ai/blog/groq-ai-chips-vs-nvidia?ref=odinhalvorson.com.

[35] Jarvislabs, "NVIDIA H100 Price Guide 2024: Detailed Costs, Comparisons & Expert Insights," *Jarvislabs.ai*, Apr. 12, 2025. https://docs.jarvislabs.ai/blog/h100-price